CMST 20(4) 113-127 (2014) DOI:10.12921/cmst.2014.20.04.113-127

Molecular Dynamic Simulations of a Simplified Nanofluid

A. Sergis*, **Y. Hardalupas**

The Department of Mechanical Engineering
Imperial College London, London SW7 2AZ, UK
**E-mail: a.sergis09@imperial.ac.uk*

Received: 04 August 2014; revised: 06 November 2014; accepted: 17 November 2014; published online: 30 December 2014

Abstract: This study describes the methodology that was developed to run a Molecular Dynamics Simulation (MDS) code to simulate the behaviour of a single nanoparticle dispersing in a fluid with a temperature gradient. A soft disk model described by the Lennard-Jones potential is used to simulate the system. The nanoparticle is assembled via the use of four subdomains of interatomic interactions and hence presents in full resolution the transfer of energy from the fluid-to-solid-to-fluid subdomains. A cluster computing system (HTCondor) was used to perform a large scale deployment of the MDS code. The obtained showcase results were successfully evaluated using three widely documented tests from the associated literature (Randomness, Radial Distribution and Velocity Autocorrelation Distribution Functions). It was discovered that the nanoparticle travels a larger distance in the fluid than the distance travelled by a fluid molecule (recovery region). The findings were confirmed by calculating the Green-Kubo self-diffusivity coefficient halfway through the simulation at which an enhancement of 156% was discovered in favour of the Nanoparticle. This might be the physical mechanism responsible for the experimentally observed thermal performance enhancement in nanofluids.

Key words: nanofluids, nanoparticles, MDS, HTCondor, heat transfer, computational study

I. BACKGROUND

Nanofluids are binary mixtures consisting of a carrier/base fluid (usually conventional coolants) and a small volumetric concentration (ranging from sub 1% up to 10%) of solid particles with a size of usually less than 100 nm. They were invented by Choi [1] in the mid-nineties initially as an alternative to micron sized solids used inside conventional coolants to increase their thermal conductivity. The original idea was that nanoparticles instead of micron particles would create less erosion problems by abrasion on the coolant conveying channels and reduce sedimentation issues usually encountered with micron sized particles. However, adding small quantities of nanoparticles to fluids proved experimentally to be more beneficial than initially expected as a very small addition of nanoparticles leads to a high thermal enhancement over the carrier fluid, which, so far, cannot be fully explained using the classic thermodynamic models. Enhancements of the order of 9%, 10-14%, 40-44% and 100-200% were observed by the majority of researchers for the purely conductive, mixed conductive/convective, pool boiling and critical heat flux heat transfer modes, which promises a step change to the overall performance of heat transfer related applications [1, 2].

Apart from the thermal performance benefits of nanofluids over conventional fluids, secondary functions make them highly desirable by designers of thermal processes. Nanofluids provide an opportunity to custom build coolants to suit specific purposes, since the number of available degrees of freedom increases significantly relative to normal coolants. Apart from the conventional coolant chemical composition, temperature of application and flow state, nanofluids offer another set of tuneable parameters such as nanoparticle material, size, shape and concentration, all of which are expected to have a direct effect on the thermal conductivity characteristics. As such, since 2000 and onwards, nanofluid research has been increasing almost exponentially in order to understand the heat transfer mechanisms that enable their augmented thermal performance as well as quantifying the effects of the tuneable parameters on the overall heat transfer enhancement [3-18].

Previous theoretical investigations, including Molecular Dynamic Simulations (MDS) studies, have not been conclusive in providing a definitive explanation regarding the thermal heat transfer mechanisms employed by nanofluids. This has been the case as the investigators frequently modelled the problem using several assumptions regarding the nature of the fluid and the heat propagation it employs [2, 8, 9, 17-22]. Theoretical investigations usually simulate nanofluids using classical thermodynamic principles applied to a single phase modelled fluid. Various correlations regarding the heat propagation values based on previous experiments are used to tune these models. In this type of studies, there is no physical modelling of the nanoparticles themselves or their interaction with the basefluid. This approach makes it impossible to discover the real nature of heat transfer as well as to predict the general thermal performance of these two-phase fluids, since the mechanisms of heat transfer are mainly pre-determined from the domain set up and rely on correlations that might not extrapolate to the more general case. From a previous study by the current authors [2], it was discovered that there is a large diversity between experimental results obtained from various research groups around the world. This happens due to the lack of standardisation of nanofluid preparation, type and experimentation. Therefore, it becomes difficult to tune theoretical investigations by using measured thermal performance correlations generalised across the nanofluids domain, since such correlations may depend on a specific method of nanofluid preparation, type and experimentation. MDS studies come to solve this issue by resolving the nanoparticle and fluid interactions and, hence, allow the system to evolve under its own dynamics and kinematics rather predetermining it from the domain set up. These simulations require a vast amount of computational power to fully resolve the interactions involved. As such, in these studies, domain simplification is something, which is usually employed to reduce computational burden. The most usual method is by switching between the molecular and macroscopic domain, namely parts of the domain are simulated via a simple classical thermodynamics analytical model and the calculated values are then passed through to the parts of the domain simulated using molecular dynamics. Usually, simplistic analytical models are used to simulate the heat transfer through the nanoparticles, while the nanoparticles themselves are considered circular or spherical with no surface texture in the 2D and 3D cases correspondingly. Even though the inner analytical models used to simulate heat transfer through the nanoparticles are well established from classical thermodynamics (single phase heat transfer), this presents a plethora of adverse complications arising of how these solutions are coupled with the molecular dynamic parts of the domain. The transfer of heat between the nanoparticle surface and the surrounding fluid molecules is very complex. The interaction of the nanoparticle surface and the surrounding fluid, which leads to an exchange of energy, are heavily dependent on the nanoparticle surface, orientation, surface energy content resolution, texture and nanoparticle dynamics and kinematics (including nanoparticle spin). The simplified models usually assume a 2D/3D linear transfer of energy across a circular/spherical nanoparticle with no surface texture, no spin and uniform surface energy content, which in its entirety ignore these factors. This has the adverse effect of dictating an unnatural and highly simplified overall method of heat transfer, which is expected to interfere with the final results. More complex MDS simulations try to resolve this issue by creating detailed models of the atomic form and bonds of every molecule involved in the solution. These models usually require even more computational power, hence more simplifications and assumptions are still carried onwards from classical thermodynamics, which makes it very hard to prove whether each and every one of them might have an effect on the final solution. To sum up, the outcomes from theoretical and MDS studies found in the literature might have been affected by the assumed modelling processes. There is hence a necessity to break up the task into its simplest form and with the least amount of assumptions create a model detailed enough to study the heat propagation mechanisms in nanofluids from first principles.

This study aims to investigate the conductive heat transfer mode mechanism through a simplified, single particle nanofluid. This is achieved via a Molecular Dynamics Simulation code, which is composed to study the dynamic dispersion of a nanoparticle, which has a surface structure and is not assumed to be spherical, and compare this to the corresponding dispersion of a fluid molecule. Single fluid atom and nanoparticle tracking is used throughout the simulations to be able to extract the molecular path data required. This process has to be repeated thousands of times in order to obtain the statistics of the nanoparticle dispersion process with low statistical uncertainty. The code is deployed at a computational scale of the order of millions of Central Processing Unit (CPU) hours with a minimum domain size of 1600 atoms. This makes it impossible to run the code on a single conventional computer system as a vast amount of high throughput computational power is required to run thousands of simulations and be able to extract macroscopic statistical thermodynamic data from a chaotic domain behaviour. As such, a cluster computing system (HTCondor§) had to be used. The MDS model makes no assumptions regarding the macroscopic thermodynamic quantities of the fluid, while the domain set up is such that preserves the simplicity of molecular collisions. The associated macroscopic thermodynamic quantities of the system are calculated via usual statistical molecular dynamics analysis performed for each simulation run at the post-processing stage of the results. The system is chaotic in its nature; namely, any change in the energy content at each point of the system will affect its evolution. As such, the system is initiated at a random stage at every run and allowed to evolve without any thermostatic or energy regulation control on the domain. An ensemble of the results of all runs builds a statistically credible outcome for the macroscopic thermodynamic results presented. The backbone of the developed code is based on the available MDS models across the academic community (in particular from [23]).

The remaining paper describes the methodology and the deployment of the MDS code on the computational domain, which is novel in its own right. It continues with the verification of the operation of the MDS code and the presentation of initial results that propose an explanation of the physics leading to enhanced heat transfer in nanofluids. The paper ends with a summary of the main findings.

II. METHODS

This section contains the methodology followed to design and deploy the code via the HTCondor cluster.

II. 1. General domain set-up

This section describes the methodology followed to assemble, test, benchmark and run the MDS code routines.

II. 1. 1. MDS core Code

The core of the MDS code, taken from the literature [23], is written in Fortran 90 and compiled via Microsoft Visual Studio 2008 on a Windows platform (Windows XP and later on Windows 7). The simplified Lennard-Jones potential model (Eq. 1) is used to simulate the forces between liquid argon atoms in the domain (Eq. 2). Eq. 2 can be derived upon integrating Eq. 1. Eq. 3 can be derived from Eq. 2 by substitution and represents the force vector an atom experiences in the domain.

$$\boldsymbol{u}\left(\boldsymbol{r}_{ij}\right) = 4\varepsilon\left[\left(\frac{\sigma}{\boldsymbol{r}_{ij}}\right)^{12} - \left(\frac{\sigma}{\boldsymbol{r}_{ij}}\right)^{6}\right] + \epsilon, \;\; |\boldsymbol{r}_{ij}| \le r_c = 2^{\frac{1}{6}}\sigma \tag{1}$$

$$\boldsymbol{f}_{ij} = \begin{cases} \left(\frac{48\varepsilon}{\sigma^2}\right)\left[\left(\frac{\sigma}{\boldsymbol{r}_{ij}}\right)^{14} - \frac{1}{2}\left(\frac{\sigma}{\boldsymbol{r}_{ij}}\right)^{8}\right]\boldsymbol{r}_{ij}, & |\boldsymbol{r}_{ij}| \le r_c \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$m\ddot{\boldsymbol{r}}_i = \boldsymbol{F}_i = \sum_{\substack{j=1 \\ (j \ne i)}}^{N_a} \boldsymbol{f}_{ij} \tag{3}$$

where

$\boldsymbol{f}_{ij}$ – force vector between the i^th^ and j^th^ pair of atoms,
m – atomic mass,
N_a – number of atoms in the domain,
$\boldsymbol{r}_{ij}$ – distance vector between the i^th^ and j^th^ pair of atoms,
$\ddot{\boldsymbol{r}}_i$ – acceleration vector of the i^th^ atom ,
$\boldsymbol{u}_{ij}$ – potential energy vector between the i^th^ and j^th^ pair of atoms,
ε – strength of interaction,
σ – characteristic length scale,
r_c – cut-off distance at which we assume that the attractive tail of the model is no longer significant.

It can be concluded from equations 2 and 3 that the controlling dimensional parameters of the system are σ, m and ε. Non-dimensionalisation of the system is achieved by substituting r with $r\sigma$ for the units of length, e with $e\varepsilon$ for the units of energy and t with $t\sqrt{m\sigma^2/\varepsilon}$ for the units of time. As a result, the equation of motion is reduced to non-dimensional units in Eq. 4.

$$\ddot{\boldsymbol{r}}_i = 48\sum_{j(\ne i)}\left(\boldsymbol{r}_{ij}^{-14} - \frac{1}{2}\boldsymbol{r}_{ij}^{-8}\right)\boldsymbol{r}_{ij} \tag{4}$$

The model is used in its 2D form with periodic boundaries on the x and y axes forming the surface of a torus. To ensure that any wraparound effects are avoided (a paradox of having a perturbation from an atom travelling around the smallest wrap-around dimension(s) of the torus and returning to self-affect/excite the atom), the domain dimensions are chosen to be much larger than the cut-off distance of the molecules (the cut-off distance is the distance that defines the extent of the interaction force field for each molecule in the domain). The system is initialised with each atom uniformly distributed in the assigned domain area using a desired density and initial system temperature. The initialisation process aims, firstly, to define the average intermolecular spacing, according to user-selected number of atoms and density, and, secondly, to assign random velocity components to each atom, thus ensuring that the system depicts a collection of atoms at a given time. An equilibration process is applied only once on the first iteration uniformly across the domain and after the velocity vector assignment to ensure that the system is initialised at the given initial temperature. The system is subsequently left to reach a steady state without any more invasive controls on the energy content of the system. The integration of the equations of motion is solved numerically using the "leapfrog" method.

The domain is initialised with a fixed density of 0.8 Non Dimensional Units (NDU) and a temperature of 1 NDU across the simulations performed. This translates to a square domain size length (L) according to Eq. 5 and an initial indicative velocity magnitude sum for the first iteration, used for equilibration, given by Eq. 6.

$$L = \sqrt{\frac{N_a}{\rho}} \tag{5}$$

$$T = \frac{1}{dN_a}\sum_i \boldsymbol{v}_i^2 \tag{6}$$

where

d – number of dimensions (here $d = 2$ since a 2D system is used),

T – initial temperature of the system,

ρ – overall number density of system.

The system reaches steady state in this configuration after the 5000^{th} iteration, which corresponds to a Non Dimensional (ND) time of 25 units. A time step of 0.005 NDU is selected, which for argon liquid atoms corresponds to a real time of approximately 10^{-14} seconds – a typical value used for MDS simulations to ensure stability and accuracy. The link between the theoretical MDS calculations presented in this study and a real ensemble of monoatomic molecules will be that of liquid argon.

Soft disks are used to represent the fluid molecules in this investigation while the fabrication of the nanoparticle will also translate to an assembly of these disks with altered force relations. This path is preferred, instead of modelling the exact chemistry of nanofluids, as the domain modelling remains simple and requires fewer assumptions to set up which might, in their turn, affect the overall solution.

II. 1. 2. MDS Code extensions – boundaries and temperature gradient

The core model represented an infinite system, namely a system at which there are no boundaries in any dimension as those are handled by a wraparound function. It was desired to use a semi-infinite system with one dimension being infinite (horizontal x-axis direction), while bounding the system in the vertical dimension (y-axis). This system transformation in space can be envisaged by a shape change from a full toric surface into a cylindrical surface. This was required in order to be able to apply a finite temperature gradient in a bound coordinate and investigate the molecular motion across it without the implications of a wraparound coordinate.

"Walls" are created at the extremes of the y-coordinate. The walls are stochastic, which means that the walls do not have any physical constituency (they are not represented by a fixed chain of atoms or forces at a given location), but an arbitrary theoretical one. The theoretical location of the wall is marked (i.e. the end of a half domain length with the coordinate system centred at the geometrical centre of the domain). The code will detect if any atoms are about to cross the imaginary wall boundary and before updating their location during the next iteration, their location and their y-velocity components will be corrected so as to reflect off the "wall" and return into the domain. This is a hard-wall correction, which does not represent a true physical process and hence leads to minor energy leakages. Nevertheless, it is a simple and typical model used for this type of applications, which will allow heat to be extracted or added to the system.

A linear temperature gradient is achieved across the y-direction by regulating the "temperature" of the walls. Keeping the temperature of the lower wall to 1 NDU, it is possible to induce a temperature gradient in the system by increasing the temperature of the top wall. The regulation of wall temperature is achieved by the hard wall routines. Every time an atom reflects on the fixed boundaries, its velocity magnitude components can be altered to represent an isotropic energy transfer to the atom. The history of the velocity magnitudes prior to reflection is logged and an indicative "single atomic temperature" purely based on the kinetic energy of the atom is calculated. If the temperature is different than the prescribed "wall temperature", the code decides if the atom will lose energy (atom hotter than wall) or gain energy (atom is colder than wall) to equilibrate the atomic temperature with the wall temperature during a wall collision. The energy transfer is performed isotropically by matching the overall atomic magnitude to the wall temperature, while preserving the reflection angles and hence the angles of the reflected velocity component.

II. 1. 3. Nanoparticle assembly and handling processes

Two approaches were followed to assemble the nanoparticle. The first approach – which required less computational power and complexity – was the "oil droplet" model. This model was developed to allow the initial formation and validation of the routines without the need of a supercomputer. This is achieved by limiting to a large extent the detail of interatomic interactions. Following the validation of this code extension, it was possible to advance to the final "solid particle" model [24] at which a more elaborate procedure is followed to assemble the nanoparticles, while the computational time for the simulations increases significantly. The final assembly routines are based on [24]. The theory behind the two models for nanoparticle assembly is described below, which have been selected in order to generate a realistic nanoparticle, which does not have a spherical shape.

(a) The "oil droplet" model

The initial attempts to form a more crystal like particle followed a simple "oil droplet" approach. An atom was selected from the domain and its interatomic attraction value ε was increased from the nominal 1 NDU to a larger value. This enabled the formation of more rigid bonds between the Argon atoms forming the oil droplet. The larger the ε value, the greater the effects of bond strength variations around the atom. This model was simple and did not require much computational power to conclude hence, the initial model of nanoparticle assembly tested and helped forming the final extensions for the MDS computation and the post-processing codes.

The model, however, had limitations, which required a more detailed approach. The main limitations arising from the preliminary model were the inability to form a crystallic solid with a fixed size as well as performing any solid-solid and solid-fluid interactions. Cross-correlation routines (Eq. 7) were used to define the boundaries of the oil droplet.

$$\mathrm{CCR}(r_i) = \hat{\boldsymbol{v}}_i \cdot \hat{\boldsymbol{v}}_{NP} \tag{7}$$

where

$\hat{v}_i$	– unit vector of the i^{th} atom,
$\hat{v}_{NP}$	– unit vector of the central atom of the assembled nanoparticle,
$\mathrm{CCR}(r_i)$	– cross correlation function value for the molecule "i" located at a distance r from the central "oil droplet" molecule.

A spatial velocity cross correlation function was used to correlate the velocity field of the centre of the "oil droplet" to the surrounding molecules at a fixed time. The $\mathrm{CCR}(r_i)$ gives peaks of the directional vector matching of the nanoparticle centre velocity vector with that of the neighbouring atoms. The $\mathrm{CCR}(r_i)$ ranges from -1 to +1, with +1 being a case of perfect velocity directional matching and -1 an antiphase velocity vector directional matching of 180°. The peaks of the $\mathrm{CCR}(r_i)$ are expected to mark the atoms, which on average move in the same direction as the central nanoparticle atom, hence identifying a cluster formation. A lower threshold peak value magnitude of less than 1 was used to acknowledge a match in order to allow for small directional mismatching due to the motion of atoms inside the oil droplet domain. Other quantities apart from the unit vectors were also introduced in parallel (such as velocity magnitude and local temperature matching) using simpler approaches than Eq. 7; however, the correlation functions gave small and indistinct peaks. This indicated that the oil droplet formed was highly volatile and unstable with no defined boundaries in between the droplet and the rest of the fluid atoms, as there was an absence of the needed intermolecular forces and interactions to create them. The shape and mass of the droplet was changing continuously from collisions with other fluid atoms or the walls while the fluid-oil droplet interactions were performed similarly to fluid-fluid interactions in order to simplify the scripts and reduce the required computational power.

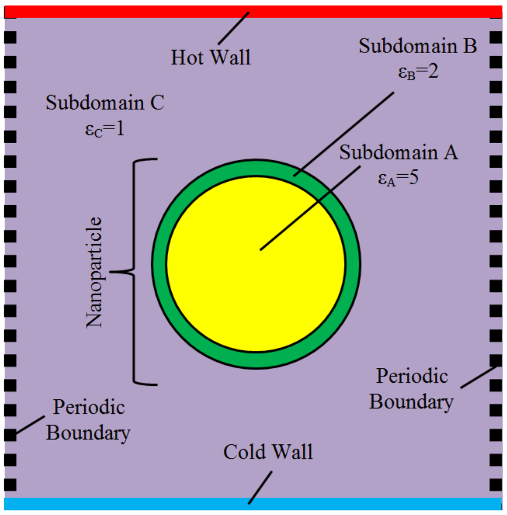

Fig. 1. Domain set up schematic indicating main features

(b) The "solid particle" model

The limitations of the "oil droplet" model led to the development of a more elaborate, computationally demanding and highly complex assembly approach, based on the 'solid particle' model [24]. The solid particle is assembled by the formation of 3 subdomains inside the main domain, as shown in Fig. 1, which are the core of the particle, subdomain A, the surface of the particle, subdomain B, and the surroundings of the particle, where the fluid argon atoms exist, subdomain C. The atoms in each subdomain are assigned a different value for their interatomic potential force parameter ε. Subdomain A atoms have an ε_A of 5 NDU, subdomain B an ε_B of 2 NDU and subdomain C an ε_C of 1 NDU (as commonly used in other MDS simulations [24]). A surface domain is required to limit "evaporation" effects occurring by collisions on the surface of the particle and loss of mass.

A radial selection of atoms is executed to assemble the nanoparticle. This is done by selecting an atom, which is regarded as the central atom of the nanoparticle from which the subdomains are radially and concentrically created according to the considered nanoparticle size (subdomain B is kept at a fixed width of 1 NDU). If an atom changes subdomains throughout the simulation, then its ε parameter is automatically updated to match that of the prescribed subdomain. Eq. 4 is altered to accommodate the effective interatomic attraction parameter ε_{eff} (equations 8 and 9) of the combination of forces involved by the interaction of atoms in different subdomains with each other. There are 3! force combination pairs involved (A-A, B-B, C-C, A-B, A-C, B-C as denoted by their subdomain characterisation letters) arising from the various subdomain characteristics and according to the given location of the atomic pairs i-j under consideration.

$$\ddot{\boldsymbol{r}}_i = 48\varepsilon_{\text{eff}} \sum_{j(\neq i)} \left(\boldsymbol{r}_{ij}^{-14} - \frac{1}{2}\boldsymbol{r}_{ij}^{-8} \right) \boldsymbol{r}_{ij} \tag{8}$$

$$\varepsilon_{\text{eff}} = \varepsilon_i \varepsilon_j \tag{9}$$

This approach assembles nanoparticles, which are stable, while the surface evaporation is limited by the use of a surface subdomain., An automatic replenishment of lost nanoparticle constituent atoms is performed during each iteration, in order to ensure that the generated nanoparticles retain their size and mass. A crystal-like core is expected to be formed (subdomain A), which still allows limited motion between the bound atoms thus simulating more realistically the energy transitions through the nanoparticle.

II. 1. 4. Evaluation of developed MDS code

The evaluation of the developed MDS code was performed using data from [23]. The data included system properties such as temperature (Eq. 6), kinetic energy (Eq. 10), potential energy (Eq. 11), total energy (sum of kinetic and potential energies of the system) and system pressure (Eq. 12).

Convergence tests were additionally performed to detect system instabilities on the same quantities using different running durations of the simulation.

$$E_k = \frac{1}{2N_a} \sum_{i=1}^{N_a} \boldsymbol{v}_i^2 \qquad (10)$$

$$E_u = \frac{4}{N_a} \sum_{1 \le i < j \le N_a} \left(\boldsymbol{r}_{ij}^{-12} - \boldsymbol{r}_{ij}^{-6} \right) \qquad (11)$$

$$P = \frac{1}{A} \left[N_a T + \frac{1}{d} \left\langle \sum_{i=1}^{N_a} \boldsymbol{r}_i \cdot \boldsymbol{F}_i \right\rangle \right], \qquad (12)$$

where
E_k – kinetic energy of system,
E_u – potential energy of system,
P – system pressure.

For the heated system, it was discovered that with all of the core extensions in place the system properties reach a steady state on average after 20-30 thousand iterations, which corresponds to a ND time of 100-150 units (where the temperature of the system reaches a plateau). Without any energy input (hot wall temperature the same as cold wall temperature), a significant energy leakage was present. The energy leakage was following an exponential trend with the system being drained of 50% of its initial total energy after 100 thousand iterations and 98% of its total energy after 1 million iterations. The leak was compensated for and the system reached steady state with the application of a temperature surplus compared to the initialisation state as small as 0.16 NDU on the hot wall (for comparison the minimum temperature difference used for the parametric studies performed was 1 NDU). The leakage is arising from the implementation of the hard wall model.

The validity of the system with the addition of the extensions was also verified via two test processes. The first was a comparison of the randomness distribution functions (Eq. 13, Fig. 2) [25, 26], which provides information regarding the atomic distribution in the domain. The domain is sub divided into regions (grid division) and the atomic number inside each grid is compared to the expected one in order to form an atomic distribution map. This test was performed without the nanoparticle assembly routines active as the system atomic distribution is altered with the formation of nanoparticles hence comparisons with the reference cases (core code with no extensions) is no longer possible. A good agreement with the reference core code was achieved after the extensions were activated, hence it can be concluded that the extensions have no effect on the normal operation of the domain.

$$D = \frac{\sigma_{\text{grid}}^2 - \lambda_{\text{grid}}^{\frac{1}{2}}}{\lambda_{\text{grid}}}, \qquad (13)$$

where
D – randomness function value,
σ_{grid} – standard deviation of atomic numbers in each grid cell counted,
λ_{grid} – expected number of atoms in each grid cell.

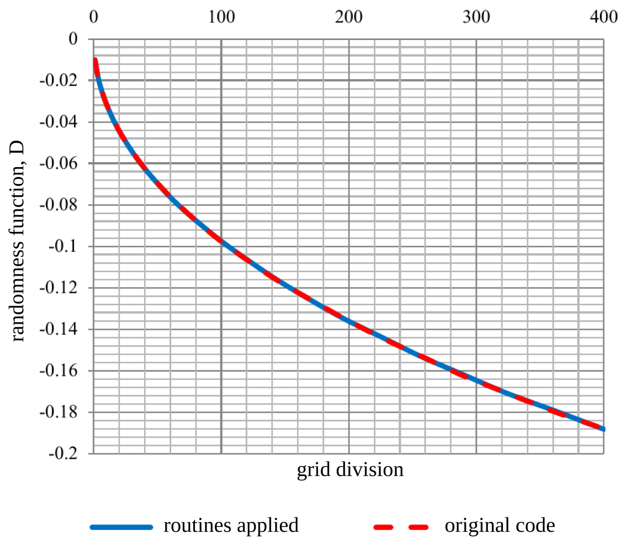

Fig. 2. Comparison of the Randomness distribution functions for changing grid division in a 10 000 particles domain after 100 000 iterations with and without the extensions

The radial distribution function (RDF) was also employed as a validation test. This function provides information regarding the atomic concentration ring bands as seen by an atom at their centre (Eq. 14).

$$g\left(r\right) = \left(\frac{N\left(r\right)}{\Delta V\left(r\right)} \right) / \sigma, \qquad (14)$$

where

$g(r)$ – radial distribution function value of the rth ring from the central atom,
$N(r)$ – number of atoms in the rth ring from the central atom,
$\Delta V(r)$ – elemental volume (corresponding ring area in a 2D domain) of the rth ring from the central atom.

The peaks and crests of the function, their spacing and amplitude provide a full profile regarding the structure of the domain (all of the thermodynamic quantities of the system can be resolved using the RDF function). The results (Fig. 3) were in agreement with similar ones found in the literature for the Lennard-Jones model [23, 24, 27] for the "calibration" RDF function plot, where the system is used with all but the nanoparticle assembly MDS code extensions activated. For the "Nanoparticle" RDF function plot (Fig. 3), all the

MDS code extensions are activated and the RDF function is similar to the RDF functions found in the literature regarding crystallic solid dendrimers in fluid/gas mixtures [24, 27-29]. An averaging process of 1000 iterations is used to assemble the RDF functions for both samples. It is evident that the process used to assemble the Nanoparticle is creating a compound structure arrangement resembling that of a solid. The process of nanoparticle assembly involves "freezing" the selected atom arrangement and imposing a crystallic intermolecular force profile. As a result, this produces a crystalline structure with less density than the surrounding fluid Argon atoms, as indicated in Fig. 3. The structure has similar RDF profile to that of a dendrimer. The RDF tends to a value, which is slightly sub-one NDU for longer radial distances, indicating a marginally reduced density (of about 5%) compared to the theoretical one. This is due to the small deficit of molecules left in the fluid domain, because a number of them were used for the Nanoparticle assembly. The nanoparticle for this investigation had a radius of 7 NDU of length, which corresponds to a mapped real nanoparticle size of 2nm. A fixed temperature gradient, for both cases, of 0.02 NDU of temperature per NDU of length is also imposed.

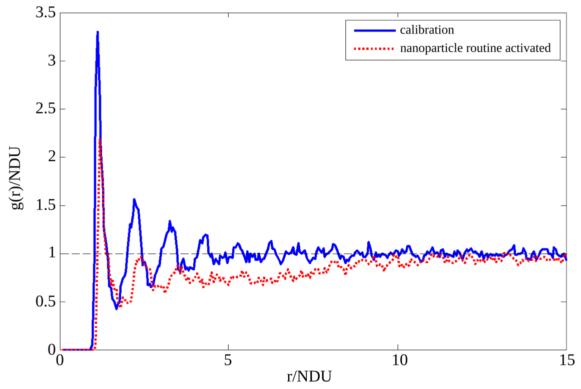

Fig. 3. Comparison of Radial Distribution Functions (RDF) plot for the Calibration and Nanoparticle cases

The mapping of the nanoparticle size is performed by comparing the relative sizes between the atoms/molecules forming a nanoparticle and its carrier fluid in a real nanofluid and applying this size comparison between the fluid atoms and the assembled nanoparticle in the MDS. For this study, a size mapping relation corresponding to a gold-water nanofluid is applied, since the relevant gold to water molecular size ratio matches the relevant nanoparticle to fluid atom sizes used in the MDS code.

The final test performed was to investigate the trends of the normalised Velocity Autocorrelation Function (VACF) for the nanoparticle and compute the self-diffusion coefficient to verify its agreement with the literature (Eq. 15). The normalised Velocity Autocorrelation Function (VACF_N) is described by Eq. 16 and the self-diffusion coefficient can be calculated by the Green-Kubo relationship – Eq. 17.

$$\mathrm{VACF} = \langle V(t_1) \cdot V(t_1 + t') \rangle\,, \quad t_1 \le t' \le t_2 \tag{15}$$

$$\mathrm{VACF}_N = \left\langle \frac{V(t_1) \cdot V(t_1 + t')}{|V^2(t_1)|} \right\rangle, \ t_1 \le t' \le t_2 \tag{16}$$

$$D_{\text{self}} = \frac{1}{3} \int_{t_1}^{t_2} \langle V(t_1) \cdot V(t_1 + t') \rangle \, dt', \tag{17}$$

where

D_{self}	– self-diffusion coefficient
t_1, t_2	– reference times where the diffusivity investigation is initiates (t_1) and ends (t_2)
t'	– integral time variable
V	– atomic velocity vector

The domain has a size of 56×56 atoms with a temperature gradient of 0.02 NDU of temperature per NDU of length. The results are averaged across 1385 complete runs of the code (simulations) with a fixed simulation duration of 100k iterations. The number of runs used was the minimum to be able to provide a credible statistical outcome for the presented values.

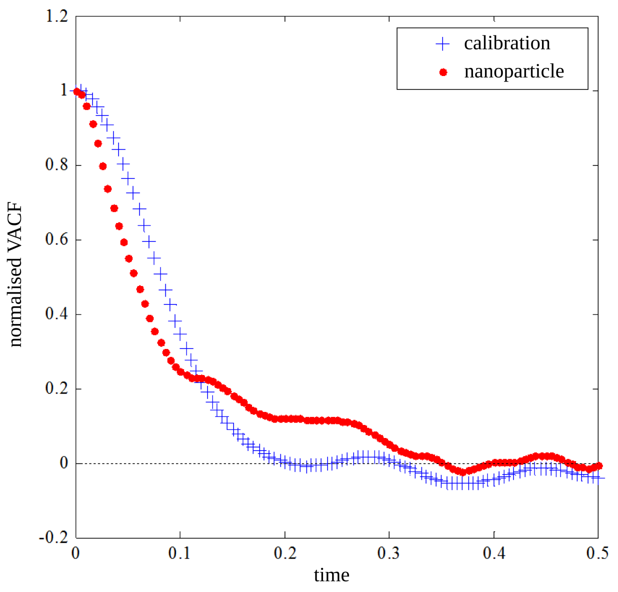

Fig. 4. Comparison of the normalised VACF for the calibration and Nanoparticle cases

Fig. 4 displays a normalised VACF plot for the first 0.5 out of the 5 ND time units considered in the investigation for the 50$^{\text{th}}$ iteration step (about half way through the simulation corresponding to a ND time of 250). Two data sets are plotted, one without a nanoparticle formation (calibration) and one with a nanoparticle of 2 nm in diameter in place. Both data sets indicate an exponential decay of the normalised VACF, which is in agreement with the literature [24, 30-32].

The self-diffusion coefficient for each iteration step was computed by using the trapezium rule with a linear unit step interval to calculate the integral of the VACF from a translated time of zero ND time units (t_1) up to the point where each plot first crosses the x-axis.

II. 1. 5. Randomisation process

Care was required to include suitable randomisation process that will allow the extraction of results as well as enabling the debugging process. During the design and validation of the code, the randomisation is required to create a random initialisation state of the system in time, which should be repeatable. This was achieved by fixing the native randomisation input argument (t_i) and hence setting up a random but frozen initialisation state of the system in time (state). The evolution in time of identical systems (simulation running time t_{run}), given the identical initialisation arguments (t_i), was the same; hence it was possible to study the effects each routine imposed on the system by studying the evolutionary system changes (state) compared to the standard basic timeline evolution of the domain (baseline).

In order to extract realistic statistical results on macroscopic thermodynamic phenomena from the code, a more elaborate randomisation process of the initialisation state needed to be performed. In the real world, the atoms in a domain are continuously moving (unless the domain is frozen to either its absolute zero temperature or in time) in a chaotic manner. Hence the system (state) is different at every instance (t_i chosen to initiate the experiment must be a unique number). If it was possible to record full resolution snapshots of a thermodynamic system through time, the probability of getting exactly the same system state (same molecular speed, location and direction of every atom) as one in the past is impossible considering realistic timescales. The monotonic variable that gives rise to the random arrangements of the system in the real world is the time dimension (t_{real}). As such, in order to recreate the various system states that will model a realistic process, the randomisation process has to reproduce a different and unique initial state for each complete simulation under account. This was achieved by seeding the native randomisation routine of the code by a unique number (t_{real}), which links the domain time evolution (state$_i$) to the real time dimension. A diagram of the process described in this section can be found in Fig. 5.

The seeding process is using the high resolution timing facility of Windows 7 environment, which has a time resolution of 100 ns to retrieve the date-stamp (t_{real}) of the execution of the code (date and time are included). Each job is assigned sequentially to each core processor with a maximum period of job submission of the order of at least 30 ms (namely t_{real} is always a unique number which increases monotonically). The core processors are synchronised with the Windows 7 environment time servers, hence it is assumed that they are all following the same timeline reference. It is hence possible to initialise the domain at a unique state (state$_i$) for the number of simulations performed for each parametric study.

II. 2. HTCondor environment set up

It was decided to employ a novel technique to run the vast amounts of simulations required for this study. This was based on HTCondor, which operates by taking advantage of the idling time of networked processors. In this case study, processing cores around the Imperial South Kensington campus belonging to networked computers, normally used for teaching, were employed to run the vast amount of calculations required. HTCondor was used to organise, distribute, run and monitor the calculations as well as collect the resulting data across the network back to an administration cluster built for this purpose. Thousands of CPU cores were successfully employed for this study. The current authors believe that the development of this path will enable institutions to proceed with high CPU load simulations using cheaper and more user approachable platforms than the ones traditionally used (Supercomputing clusters), which might be otherwise out of reach.

More information on the HTCondor set up including hardware set up for the administration cluster can be found in Appendix A.1.

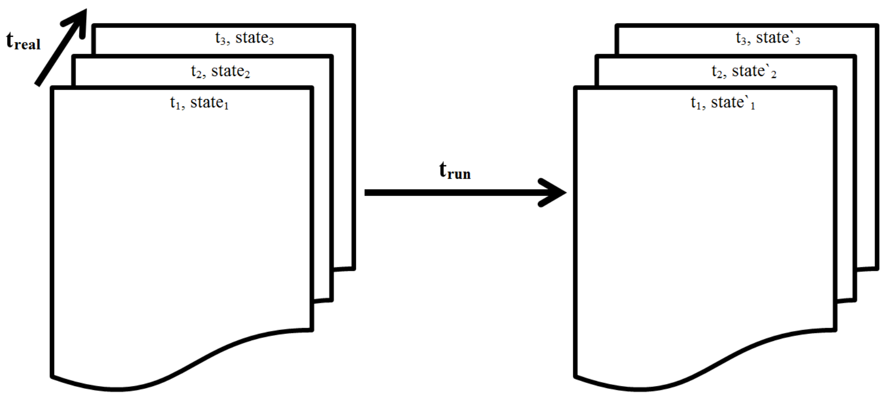

Fig. 5. Diagram of randomization routines process followed

III. RESULTS AND DISCUSSION

Extended post processing and assessment of results was implemented through Matlab scripting. This provides a wide flexibility of tools, which are not available or are hard to employ using Fortran 90. This section presents the preliminary physical results of the code. System performance results under HTCondor operation can be found in Appendix A.3.

III. 1. Visualisation of the system

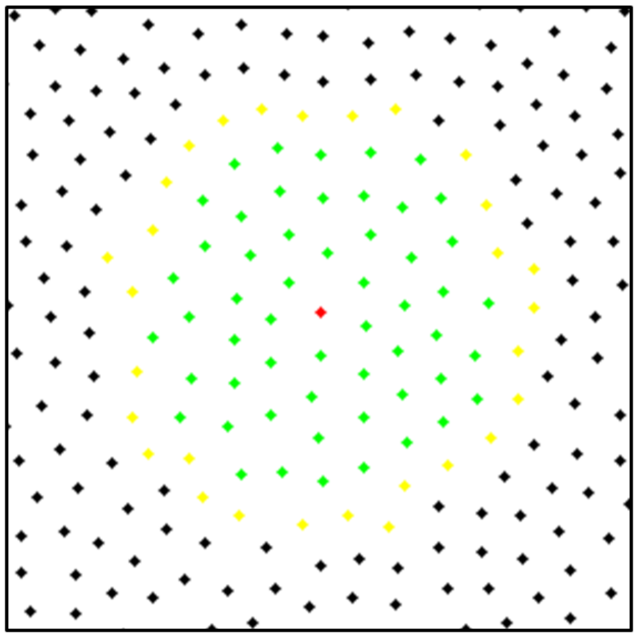

Fig. 6. Snapshot of nanoparticle assembly (fluid atoms with black colour, nanoparticle surface atoms with yellow colour, internal nanoparticle atoms with green colour and central internal nanoparticle with red colour)

The visualisations of the fluid molecules and nanoparticle behaviour were implemented through Matlab that aided code development. Fig. 6 displays an example from a snapshot of a nanoparticle assembly under the "solid particle" model. Black dots represent fluid atoms, yellow dots represent nanoparticle surface atoms, green dots represent internal nanoparticle atoms while the red dot represents the central internal nanoparticle atom. It is evident that the nanoparticle structure is modelled in full resolution, hence the properties controlling energy transfer via dynamic and kinematic processes can be modelled exhaustively. Fig. 7 shows an example of a trajectory recorded for a nanoparticleThe main postprocessing approach uses locational data from the trajectory of the central atom inside the nanoparticle as well as the current domain set up of each data set and compares it to the trajectory data of a fluid atom released from the same location as the nanoparticle without the activation of the nanoparticle assembly routine. This is in order to extract the required path statistics for the reference baseline atom and the nanoparticle. Compression algorithms had to be developed and applied through Matlab to cope with the large volume of data collected. More information can be found in Appendix A.2.

III. 2. MDS showcase results

The results from an example run are included in this section as a showcase for the MDS, Matlab and HTCondor scripts and hardware. The study focuses on a 3136 atom domain (56×56) with an applied temperature gradient of 0.02 NDU of temperature per NDU of length. The nanoparticle has a radius of 7 ND length units, which corresponds to a physical mapped size of 2 nm (diameter).

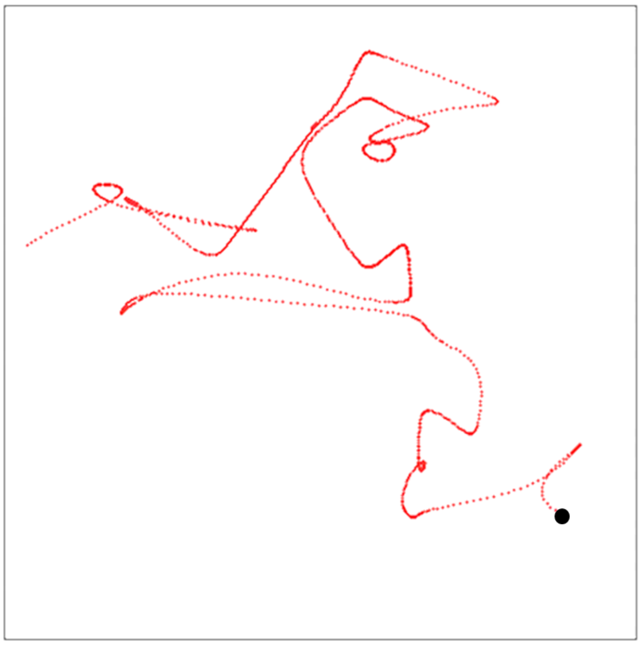

Fig. 7. Nanoparticle (represented by black dot) visualisation in time lapse (one red dot per iteration) performing Brownian motion

Fig. 8 shows a plot of the average one dimensional distance vector (against the temperature gradient) followed by a fluid atom and a nanoparticle both released from the same domain location (middle of the domain). The 95% confidence intervals from the statistical analysis indicate that the minimum number of simulations selected for this study is sufficient to attain a clear distinction between the distance vector travelled for each, the fluid atom and nanoparticle, against the temperature gradient. Initially, an oscillation exists in the first steps of the calculation (Fig. 8). The oscillation is not a result of system instabilities by the assignment of the force fields arising from the initialisation process of the system as the time scale associated is longer than the time, indicated by [23], which is required for the system to reach equilibrium. The oscillations appear to be a thermodynamic phenomenon associated with the thermal shock experienced

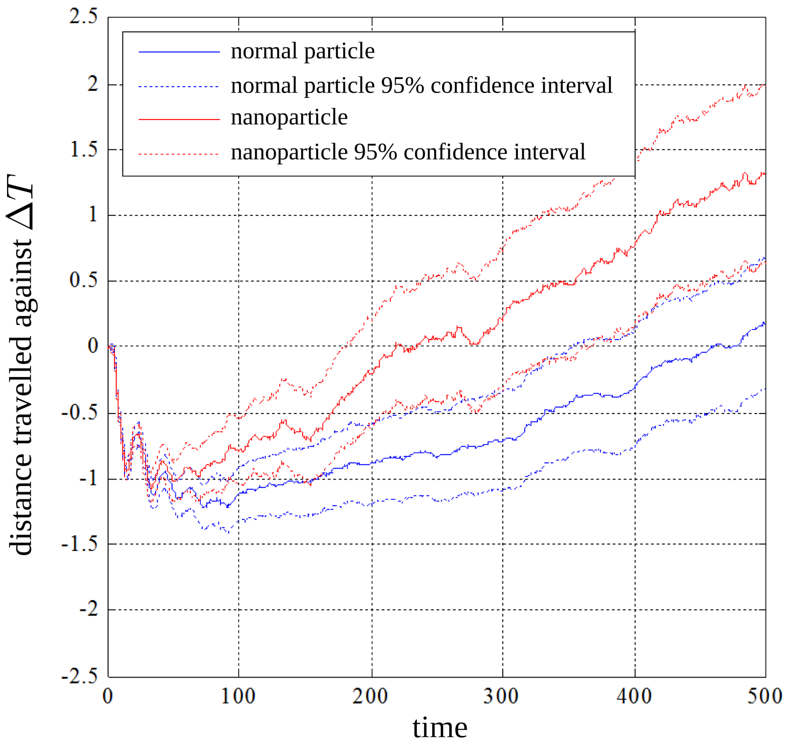

Fig. 8. Distance vector plots for a normal particle (fluid argon atom) and a 2 nm nanoparticle released from the same location in a domain with a fixed temperature gradient

by the application of the hot and cold walls and the waves of energy transfer experienced by the initially cold domain. The instabilities last for about 80 time units. During that time, the nanoparticle and fluid atom appear to have no difference in their one dimensional distances covered against the temperature gradient. The initial instabilities appear to provide a "kick" for both the fluid atom and the nanoparticle towards the colder region of the domain, which on this initial study appear to agree with the thermophoretic force expected to be present at such scales.

From 80 time units and onward, the plot of Fig. 8 indicates a clear distinction between the average behaviour of a nanoparticle and a fluid atom. A region where the nanoparticles and fluid atoms begin moving along the direction of the temperature gradient exists (the gradient of the distance plot becomes positive). The authors can only speculate that this distinction might arise due to mass diffusivity phenomena becoming more dominant than thermophoretic effects in this part of the time domain evolution. In this "recovery" region, it is yet unclear what might be the true reason, which induces a faster recovery of the nanoparticle compared to the fluid atom.

The self-diffusion coefficient for the calibration and nanoparticle cases as described in section 2.1 was found to be 0.0472 and 0.1208 ND length units squared over ND time units respectively. The self-diffusion coefficient was calculated for the complete set of simulations and statistical error was found to be negligible in its calculation. For the given interval, the self-diffusion coefficient of the Nanoparticle is 156% larger compared to a normal fluid atom in the domain, which might be accounted for the thermal phenomena observed in a real nanofluid.

It is yet unclear what the initial wave shaped instabilities are from a non dimensional time of 0-75 time units and also what causes the faster nanoparticle recovery compared to the fluid atom from a non dimensional time of 80 and onwards. Parametric studies are required to provide more definitive answers to explain the observed behaviour.

IV. CONCLUSIONS

A Molecular Dynamic Simulation code has been composed to model a simplified nanofluid containing a single nanoparticle dispersing in a prescribed temperature gradient. Three common methods were used to evaluate the resulting MDS code – the randomness distribution function, the radial distribution function and the velocity autocorrelation function. The results were compared with those found in the literature and were in good agreement. The main outcomes of the study are:

1. The study successfully demonstrated the use of a more financially affordable platform (HTCondor) used to execute this type of simulations, which may potentially assist similar, high computationally demanding studies in the future.
2. Showcase results for a 2 nm nanoparticle in a 56×56 atom domain with 0.02 ND temperature gradient units are obtained. The results of the one dimensional distance vector indicate the following:
3. Oscillations appear to be present for the initial part of the solution speculated to arise from a thermodynamic effect during the transient heating up state of the system.
4. Thermophoresis is speculated to be dominant for the first part of the solution (0-80 time units), while there is no distinction between the nanoparticle and fluid atom vector distance covered for the nanoparticle size investigated.
5. On average the nanoparticle appears to cover more distance compared to a fluid atom (baseline atom) during its motion along the temperature gradient – recovery (from a time of 80 NDU and onwards), where mass diffusivity effects appear to be more dominant.
6. The self-diffusion coefficient for the nanoparticle (Green-Kubo self-diffusivity relationship) is enhanced by 156% over that of the baseline fluid atom. This might explain the thermal enhancement observed in nanofluids.
7. Parametric investigations are required to help understand the observed behaviour.

List of abbreviations

CPU	– Central Processing Units
MDS	– Molecular Dynamics Simulation
ND	– Non Dimensional
NDU	– Non Dimensional Units
RDF	– Radial Distribution Function
SSD	– Solid State Drive
VACF	– Velocity Autocorrelation Function

Acknowledgements

We are grateful to Prof. David Heyes for his guidance and help throughout this investigation. We would like to thank our collaborator Mr. Steve Rochford from the central ICT division of Imperial College London for his valuable input in setting up HTCondor as well as the ICT manager Mr. Tim Machin who provided the required permissions to deploy the system at the South Kensington Campus. We would also like to thank the local IT staff of the Mechanical Engineering Department, Dr. Nadiri Fusun and Mr. Ben Collinson for all of their associated help in setting up the administration cluster hardware and network accesses. We would also like to thank Dr. Tamer Zaki and Dr. Edwart Smith for providing a version of Rapaport's core code and all of their associated help. This work was funded by the RCUK Energy Programme and EURATOM through CCFE. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] S.K. Das, S.U.S. Choi, W. Yu, T. Pradeep, Nanofluids: Science and Technology. John Wiley & Sons, Inc. NJ, USA 2007.

[2] A. Sergis, Y. Hardalupas, *Anomalous heat transfer modes of nanofluids: a review based on statistical analysis*, Nanoscale Research Letters **6**, 391 (2011).

[3] Y. Xuan, Z. Yao, *Lattice Boltzmann model for nanofluids*, Heat and Mass Transfer **41**, 199-205 (2005).

[4] P. Warrier, A. Teja, *Effect of particle size on the thermal conductivity of nanofluids containing metallic nanoparticles*, Nanoscale Research Letters **6**, 247 (2011).

[5] L. Vasiliev, E. Hleb, A. Shnip, D. Lapotko, *Bubble generation in micro-volumes of "nanofluids"*, International Journal of Heat and Mass Transfer **52**, 1534-1539 (2009).

[6] S.C. Tzeng, C.W. Lin, K.D. Huang, *Heat transfer enhancement of nanofluids in rotary blade coupling of four-wheel-drive vehicles*, Acta Mechanica **179**, 11-23 (2005).

[7] S. Soltani, S.G. Etemad, J. Thibault, *Pool boiling heat transfer performance of Newtonian nanofluids*, Heat and Mass Transfer **45**, 1555-1560 (2009).

[8] J.N.N. Quaresma, E.N. Macedo, H.M. Da Fonseca,H.R.B. Orlande, R.M. Cotta, *An Analysis of Heat Conduction Models for Nanofluids*, Heat Transfer Engineering **31**, 1125-1136 (2010).

[9] S.M.S. Murshed, K.C. Leong, C. Yang, *A combined model for the effective thermal conductivity of nanofluids*, Applied Thermal Engineering **29**, 2477-2483 (2009).

[10] S.M.S. Murshed, K.C. Leong, C. Yang, *Investigations of thermal conductivity and viscosity of nanofluids*, International Journal of Thermal Sciences **47**, 560-568 (2008).

[11] H. Kim, M. Kim, *Experimental study of the characteristics and mechanism of pool boiling CHF enhancement using nanofluids*, Heat and Mass Transfer **45**, 991-998 (2007).

[12] P. Keblinski, R. Prasher,J. Eapen, *Thermal conductance of nanofluids: is the controversy over?* Journal of Nanoparticle Research **10**, 1089-1097 (2008).

[13] J.-Y. Jung, J.Y. Yoo, *Thermal conductivity enhancement of nanofluids in conjunction with electrical double layer (EDL)*, International Journal of Heat and Mass Transfer, **52**, 525-528 (2009).

[14] K.S. Hwang, S.P. Jang, S.U.S. Choi, *Flow and convective heat transfer characteristics of water-based Al_2O_3 nanofluids in fully developed laminar flow regime*, International Journal of Heat and Mass Transfer **52**, 193-199 (2009).

[15] F. Duan, D. Kwek, A. Crivoi, *Viscosity affected by nanoparticle aggregation in Al_2O_3-water nanofluids*, Nanoscale Research Letters **6**, 248 (2011).

[16] Y. Ding, H. Alias, D. Wen, R.A. Williams, *Heat transfer of aqueous suspensions of carbon nanotubes (CNT nanofluids).* International Journal of Heat and Mass Transfer **49**, 240-250 (2006).

[17] W. Yu, S.U.S. Choi, *The role of interfacial layers in the enhanced thermal conductivity of nanofluids: A renovated Hamilton-Crosser model* Journal of Nanoparticle Research **6**, 355-361 (2004).
[18] W. Yu, S.U.S. Choi, *The Role of Interfacial Layers in the Enhanced Thermal Conductivity of Nanofluids: A Renovated Maxwell Model*, Journal of Nanoparticle Research **5**, 167-171 (2003).
[19] S. Jain, H.E. Patel, S.K. Das, *Brownian dynamic simulation for the prediction of effective thermal conductivity of nanofluid*, Journal of Nanoparticle Research **11**, 767-773 (2008).
[20] W. Cui, M. Bai, J. Lv, L. Zhang, G. Li, M. Xu, *On the flow characteristics of nanofluids by experimental approach and molecular dynamics simulation*, Experimental Thermal and Fluid Science **39**, 148-157 (2012).
[21] G. Chen, W. Yu, D. Singh, D. Cookson, J. Routbort, *Application of SAXS to the study of particle-size-dependent thermal conductivity in silica nanofluids*, Journal of Nanoparticle Research **10**, 1109-1114 (2008).
[22] P. Bhattacharya, *Brownian dynamics simulation to determine the effective thermal conductivity of nanofluids*, Journal of Applied Physics **95**, 6492 (2004).
[23] D.C. RapaportC, *The Art of Molecular Dynamics Simulation*, Cambridge University Press, Cambridge 1995.
[24] D.M. Heyes, M.J. Nuevo, J.J. Morales, *Self-diffusion of large solid clusters in a liquid by molecular dynamics simulation*, Molecular Physics **88**, (1996).
[25] Y. Hardalupas, S. Horender, *Fluctuations of particle concentration in a turbulent two-phase shear layer*, International Journal of Multiphase Flow **29**, 1645-1667 (2003).
[26] J.K. Eaton, J.R. Fessler, *Preferenctial Conventration of particles by turbulence*, International Journal of Multiphase Flow **20**, 169-209 (1994).
[27] M. Bamdad, S. Alavi, B. Najafi, E. Keshavarzi: *A new expression for radial distribution function and infinite shear modulus of Lennard-Jones fluids*, Chemical Physics **325**, 554-562 (2006).
[28] V.I. Korsunskii, R. Neder, K. Hradil, J. Neuefeind, K. Barglik-Chory, G. Miller, *Investigation of the local structure of nanosized CdS crystals stabilized with glutathione by the radial distribution function method*, Journal of Structural Chemistry **45**, 427-436 (2004).
[29] D.S. Wilson, L.L. Lee, *Molecular recognition and adsorption equilibria in starburst dendrimers: gas structure and sensing via molecular theory*, Fluid Phase Equilibria **228-229**, 197-205 (2005).
[30] M.H. Kowsari, S. Alavi, M. Ashrafizaadeh, B. Najafi, *Molecular dynamics simulation of imidazolium-based ionic liquids. I. Dynamics and diffusion coefficient*, The Journal of Chemical Physics, **129**, 224508-224508 (2008).
[31] V.Y. Rudyak, S.L. Krasnolutskii, D.A. Ivanov, *Molecular dynamics simulation of nanoparticle diffusion in dense fluids*, Microfluidics and Nanofluidics **11**, 501-506 (2011).
[32] V.Y. Rudyak, A.A. Belkin, *Self-diffusion and viscosity coefficient of fluids in nanochannels*, 3rd Micro and Nano Flows Conference, Thessaloniki, 22-24 2011.

APPENDIX

A. 1. HTCondor set up

HTCondor is a workload management computing system invented by the High Throughput Computing division of the Department of Computing Sciences of the university of Wisconsin-Madison (UW-Madison) around 1988 (renamed “HTCondor” from 2012 onwards from “Condor”). The system is able to take advantage of the idling time of networked computer systems (such as ordinary desktop machines) in order to run batch jobs and return the results via the local network. Such system is under trial at the South Kensington university campus of Imperial College London and in collaboration with the Information and Communication technologies (ICT) division it was decided to co-develop and test parts of the system to enable computing the lengthy MDS required for the current study. The system is installed campus wide and has a current maximum capacity of around 4000 cores with a current nominal operational capacity of 3000-3500 cores depending on usage (green saving energy higher order policies of the campus are restricting the maximum usage during the current deployment and testing phase of the HTCondor system). The machines employed to form the HTCondor bank of nodes/core processors are 64 bit Windows 7 machines incorporating multicore or/and hyper threaded processors that can be employed individually for each job assignment. The machines utilised for this purpose are the same machines located inside computational teaching cluster rooms of the campus. Automatic HTCondor script cut off (suspend and vacation routines) provide priority of node utilisation to physical users or higher order functions sent from the ICT division (e.g. for maintenance, updates etc.) over any batch job sent to the system. The HTCondor is provided under an open source license.

The current campus installation utilises the “vanilla universe” under the HTCondor 7.8.2 version. This is an operational mode, which provides flexibility for running scripts but has limited administrative services. The main limitations of the “vanilla universe” are the lack of check pointing services and remote system calling. This presents an issue for longer code runs that get interrupted and cannot be resumed on the same node as all progress is lost and the run has to be restarted. As a result, the system was not responding well to increased node user activity.

A. 1. 1. MDS Core code and HTCondor configuration file adjustment

In order to reach to the final version of the Fortran 90 MDS code for this study, significant alterations had to be performed to suit the HTCondor node environment compared to the initially developed desktop version. These changes mainly revolved around system performance increase by changing the operational profiles of the code, as well as altering the handling routines for the inputs and outputs of the code. The MDS code was compiled into an executable 64 bit Windows 7 compatible file and along with the input files it was sent by an HTCondor function to a temporary storage space on the nodes for execution. The Fortran 90 code had hence to adapt to use dynamic paths for the input and output functions as the paths changed from node to node. The code was also adjusted so as the solution was performed into short blocks, namely, it was preferred to break up the solution into shorter runs of 100k iterations than having a node running multiple simulations from the same execution file. This provides a dynamic and more reliable environment for collecting the required data as a failure to complete a short run will only cause a loss of data for the particular run instead of a series of simulations. Specific hard-kill commands are also incorporated into the MDS code to increase system performance. These routines detect when a case might become unstable and terminate it in a controlled manner in order to retrieve the results produced and free up the available nodes; something which was of great use especially during the designing and debugging process.

The HTCondor configuration files also had to be tuned to be able to increase the overall performance of the system. Taking account of the large volume of jobs required to monitor by the administration machines, the settings had to be altered from their generic profile to accommodate the specific use for the MDS code. The allocation of jobs was performed sequentially in order of the shorter to the longest runs to increase system utilisation, data retrieval and postprocessing optimisation.

A. 1. 2. Hardware requirements

The cluster was required to run a large amount of jobs (order of thousands) at any given time. This presented one of the largest challenges in setting up the system as there were many limitations on the simultaneous operation, monitoring, assignment and retrieval of the thousands of instances of the MDS code running. It was discovered that several bottlenecks existed, which halted the full deployment of the system. Real time monitoring of the nodes requires fast response from the administration machines to their corresponding nodes. A continuous communication of the nodes is required to keep track of the job assignment and completion as well as dynamically responding to node availability and priority system functions. The communication is performed by having the nodes "touching" periodically their progress files, which are located under the administration machine storage system (pulse check). By increasing the number of nodes, the responsiveness of the storage space of the administration machine appeared to be lagging to the amount of request received. This presented a hardware issue as the hard disks used, the network response and processor response of the administration was insufficient to perform these tasks. An additional problem arose from the lack of sufficient Random Access Memory (RAM) required by the HTCondor administrators to keep track of each MDS run as the amount of memory required was beyond the maximum capacity of the administration machine.

As a result, it was not possible to run more than a couple of hundreds of concurrent simulations on HTCondor from a single administration machine. It was hence decided to split up the administration of the nodes into six individual machines. Each machine was equipped with a fast Solid State Drive (SanDisk SDSSDP-064G-G25, 64GB, SATA 6GB/s, 2.5 inch Internal SSD), where the job communication files are stored and 6GB of RAM. Each administration machine is a 64 bit, dual core machine running Windows 7 with a 1Gbit network connection. A shared 1Gbit network space of 180GB for the input, execution and output files was used where each node had access to read and write hence reducing significantly the data transfer load from/to the local administrators. The current hardware provides a maximum simultaneous handling support capacity of 1200 jobs per administration machine (7200 jobs in total from the administration bank), which provides multiple redundancies for the current system operation as well as accommodating future system expandability. Each administrator is limited to a maximum number of jobs to run under normal operation according to Eq. 15. This limitation is necessary to keep the administration bank load utilisation balanced for long periods of running time.

$$M_{\text{jobs}} = \frac{N_{\text{nodes}}}{N_{\text{admin}}}, \tag{18}$$

where

M_{jobs}	– maximum number of jobs to assign on each administration machine
N_{nodes}	– number of available nodes
N_{admin}	– number of administration machines in operation

A. 2. Matlab Post Processing Routines

Traditional statistical quantities, such as means and their variances, are hence required to quantify the change in the properties investigated across every simulation and every iteration. A method of collecting statistical data from each iteration is followed across the range of the performed simulations. This led to challenges as data compression routines were required to preserve the data recorded and allow the execution of the code without overwhelming the RAM capacity of the machine running the post processing jobs. The statistics compression algorithm derived uses recursive processes, equations 16 to 20 to calculate the statistics cumulatively instead of collectively. The entire range of data is not required to estimate the final mean and variance of the population, since this is calculated incrementally. This tactic makes it possible to perform the statistical analysis with minimal memory requirements.

$$\mu_{y_i} = \frac{\sum y_i}{N_{y_i}} \tag{19}$$

$$\mu_{y_n} = \frac{1}{N_{y_n}}\left(\sum y_i + y_n\right) = \frac{1}{N_{y_n}}\left(N_{y_i}\mu_{y_i} + y_n\right) \tag{20}$$

$$\sigma^2_{y_i} = \frac{1}{N_{y_i}}\sum\left(y_i - \mu_{y_i}\right)^2 \tag{21}$$

$$\sigma^2_{y_n} = \frac{1}{N_{y_n}}(\sum y_i^2 - \mu^2_{y_n}) \quad (22)$$

$$\sum y_i^2 = N_{y_{n-1}}\left(\sigma^2_{y_{n-1}} + \mu^2_{y_{n-1}}\right) + y_i^2, \quad (23)$$

where

y	– atomic property y,
μ_{y_i}	– mean of the 1st up to the ith value of population property y,
μ_{y_n}	– mean of the 1st up to the nth value of population property y $(n > i)$,
$\sigma^2_{y_i}$	– variance of the 1st up to the ith value of population property y,
$\sigma^2_{y_n}$	– variance of the 1st up the nth value of population property y $(n > i)$.

The Matlab routines also offer automatic incremental backup, stop and resume functions in order to minimise data and computational time losses in case of a system crash, when they are implemented on a desktop machine. The routines are also adjusted and compatible to run on HTCondor should an increased load of post processing is required. The incremental backup, stop and resume functions are unavailable when they are deployed through the condor system due to aforementioned limitations of the "vanilla universe".

A. 3. HTCondor system operation and performance

The preliminary test cases investigated how the code and HTCondor environment react to long and short code runs. It is estimated that the process of designing, debugging and testing all of the systems as well as obtaining the first showcase results documented in this study to have taken an equivalent running time of about 5 million CPU hours. It was discovered that the performance of the system drops significantly for runs of the order of 16hrs and over. The performance was also moderately reduced for short runs (runs shorter than 1hr). A further investigation indicated that the bottlenecks encountered for the longer runs were due to the lack of checkpointing and network drive speed limitations upon batches of runs reaching completion. The longer runs have higher probability of getting interrupted by priority functions or users at the node systems and with the lack of a checkpointing facility the runs are cancelled and restarted.

Moreover, even if the longer runs do reach completion – even though countermeasures are taken to limit this case up to a permissible degree by randomising the write intervals after completion – there is still a high probability that a large number of nodes is simultaneously trying to transfer and write up data to a single network storage node which gets overwhelmed and discards connections. As such, the system will cancel the nodes that fail to write up the data in a pre-allocated time interval and restart them on a different node. These two antagonising factors are hampering significantly the overall system performance. Shorter runs present a similar problem however; the countermeasures in place (random assigned time delays in data transfer) as well as the distribution, allocation and running time of very short jobs reduce the impact of the problem compared to the longer runs. A maximum performance was reached for runs taking around 5hours to complete at which the problems of data transfer and the probability of simulation interruption are minimised.

The problems concerning interruptions by a priority service or a user at the node machines can be resolved by forming up a checkpointing facility. This is not offered by the native HTCondor software as an option for the "vanilla universe" running on Windows platforms. In this case, the only way to resolve the issue (without affecting the primary user functions of the nodes at the campus) is by changing the "universe" HTCondor is running the jobs into a different one, which provides more services (for example the "standard universe"). Nevertheless, this will have as a result to limit up to a large degree the flexibility the node platforms present to run scripts through the system. There is also the possibility of hard coding checkpointing functions in the "vanilla universe" however; the first attempts performed for this study indicated extensive problems with the stability of the system due to the extensive higher order campus network security and operation policies. A tertiary solution would be to abolish the singular network storage scheme and employ a clustered storage scheme with fast solid state drives (SSD). HTCondor offers this facility by disabling network storage and setting up for the files each node produces to be returned back to the administration machine. This is something to be tried in the future before larger scale changes are attempted however; this solution will impair a large CPU use of the administration machines to write the high volume and short response rate network data required which might as a result limit the overall node volume handling capacity by the administrators.

Dr Antonis Sergis graduated from the University of Cambridge in the United Kingdom with a bachelor and masters titles in Aerothermal and Aerospace Engineering (BEng, MEng, MA) in June 2009. He received his PhD degree from the Mechanical Engineering Department of Imperial College London in November 2013 on theoretical and numerical studies of heat transfer in nanofluids with an emphasis on nuclear fusion applications. His PhD was sponsored by an EPSRC, CASE award in collaboration with the Culham Centre for Fusion Energy (CCFE) – through RCUK and EURATOM. He has been employed as a Research Assistant and as a Research Associate at the Department of Mechanical Engineering of Imperial College London in November 2013 and July 2014 correspondingly. Antonis has published various studies on heat transfer in nanofluids and Nuclear Fusion cooling devices as well as contributed in many conferences and nuclear fusion related schools. His current work revolves around laser diagnostics on fuel atomisation processes in jet engine combustion chambers as well as further experimental and computational work on heat transfer in nanofluids.

Professor Yannis Hardalupas graduated from National Technical University of Athens in Greece with a degree in Mechanical Engineering in 1984 and received his PhD degree from the Mechanical Engineering Department at Imperial College in 1989, where he was appointed Professor of Multiphase Flows in 2009. He was awarded an EPSRC Advanced Research Fellowship on experimental research on combustion of liquid and solid fuels and an industrial secondment to Ricardo Consulting Engineers from the Royal Academy of Engineering for development of computational models for atomization of liquid fuels in IC engines. He is currently a member of the Committee of the Combustion Physics group of the Institute of Physics and of the British Section of the Combustion Institute, the technical committee of Propellants and Combustion of the American Institute of Aeronautics and Astronautics, the Editorial board of the International Journal of Spray and Combustion Dynamics and the Journal of Combustion and the Advisory Board of several International Conferences. In addition to his contributions in the area of combustion, liquid atomization and sprays, the development of novel optical techniques has led to patents for novel instruments on powder sizing, planar droplet sizing and nanoparticle sizing.